DOCUMENT RESUME

ED 309 762 IR 013 939

AUTHOR Swan, Karen
TITLE Programming Objects To Think With: Logo and the Teaching and Learning of Problem Solving.
PUB DATE Mar 89
NOTE 35p.; Paper presented at the Annual Meeting of the American Educational Research Association (San Francisco, CA, March 25-30, 1989).
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Hypothesis Testing; *Instructional Design; Intermediate Grades; *Intermode Differences; Models; Pretests Posttests; *Problem Solving; *Programing; Programing Languages; *Skill Development; Transfer of Training
IDENTIFIERS *Logo Programing Language

ABSTRACT

Unfortunately, much of the research devoted to Logo and problem solving has not supported the claim that Logo provides an environment in which children will develop problem solving skills, but the literature suggests that direct instruction and mediated Logo programming practice can result in the acquisition and transfer of certain problem solving abilities. The research reported in this paper was designed to test such an hypothesis by differentiating between interventions combining direct instruction and mediated practice and discovery learning approaches, and with assessing the importance of programming within that model. Subjects were 100 students in the fourth through the sixth grades who had all had at least one year (30 hours) of prior experience programming in Logo. All subjects were pretested on their ability to solve problems requiring the use of each of the five problem solving strategies under investigation, and randomly assigned by grade to one of three treatment conditions--a Logo graphics condition, a cut-paper manipulative condition, or a discovery learning, Logo projects condition. Results reveal that the model can indeed support the acquisition and transfer of four problem solving strategies--subgoals formation, forward chaining, systematic trial and error, and analogy--whereas neither discovery learning in a Logo environment nor direct instruction with concrete manipulatives practice can accomplish that. Indications are that the model can support the teaching and learning of alternative representation strategies as well. The findings support claims for the efficacy of Logo as a medium conducive to the teaching and learning of problem solving, and argue for the use of knowledge-based instructional design and computing environments in the creation of problem solving interventions. (33 references) (Author/BBM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED309762

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

PROGRAMMING OBJECTS TO THINK WITH: LOGO AND THE TEACHING AND LEARNING OF PROBLEM SOLVING

Karen Swan
SUNY Albany

paper presented at the Annual Meeting of the American Educational Research Association, March, 1989, San Francisco

IR013939

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Karen Swan

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

Abstract

The Logo computer programming language has been described as an environment in which children will develop problem solving skills. Unfortunately, much of the research devoted to Logo and problem solving has not discovered such a connection, but a careful reading of the literature suggests that direct instruction and mediated Logo programming practice can result in the acquisition and transfer of certain problem solving abilities. The research reported in this paper was designed to test such hypothesis. In particular, it was concerned with differentiating between interventions combining direct instruction and mediated practice and discovery learning approaches, and with assessing the importance of programming within that model. Results reveal that the model can indeed support the acquisition and transfer of four problem solving strategies – subgoals formation, forward chaining, systematic trial and error, and analogy – whereas neither discovery learning in a Logo environment nor direct instruction with concrete manipulatives practice can. Indications are that the model can support the teaching and learning of alternative representation strategies as well. The findings support claims for the efficacy of Logo as medium conducive to the teaching and learning of problem solving, and argue for the use of knowledge-based instructional design and computing environments in the creation of problem solving interventions.

> "Stated most simply, my conjecture is that the computer can concretize (and personalize) the formal. Seen in this light, it is not just another powerful educational tool. It is unique in providing us with the means for addressing what Piaget and many others see as the obstacle which is overcome in the passage form child to adult thinking. I believe that it can allow us to shift the boundary separating concrete and formal."
>
> – Seymour Papert (1980, 21)

Seymour Papert maintains that computers are a truly revolutionary educational medium because they support "transitional objects to think with," computer representations of abstract ideas that can be manipulated in seemingly concrete ways to help bridge the gap between concrete and formal thought. He further argues that this transition will take place automatically and painlessly when students are provided with well-structured computer environments rich in such computer manipulatives, environments such as the one he himself designed, the programming language Logo.

At least as it concerns Logo and the teaching and learning of problem solving, the literature reports no such automatic acquisition of problem solving skills. Indeed, multiple studies have reported no significant increases whatsoever in problem solving abilities among students involved in Logo programming (Papert, Watt, diSessa, & Weir, 1979; Pea & Kurland, 1984, 1987; Leron, 1985; Salomon & Perkins, 1987; Johanson, 1988). Indications are, however, that problem solving skills may be enhanced through direct instruction and mediated practice in Logo programming environments (Gorman & Bourne, 1983; Clements & Gullo, 1984; Clements, 1987; Carver, 1987; Lehrer & Randle, 1987; Thompson & Wang, 1988). Indeed, our own pilot research found that students' scores on measures of five problem solving strategies – subgoals formation, forward chaining, systematic trial and error, alternative representation, and analogy – were significantly increased following an instructional intervention that combined direct problem solving instruction with mediated practice in a Logo programming environment (Swan & Black, 1988).

The research reported in this paper was concerned with validating the success of that instructional model with respect to Papert's claims for Logo; in particular, with differentiating between that model and a discovery learning approach, and with assessing the importance of computer manipulatives within it. Two research questions were addressed:

1. Is explicit instruction and mediated practice in particular problem solving strategies superior to discovery learning for supporting the acquisition and transfer of problem solving strategies from within Logo programming environments?

2. Is the Logo programming environment particularly supportive of the acquisition and transfer of problem solving skills?

Problem Solving Strategies

A number of distinct problem solving strategies can be distinguished from general problem solving behaviors (Newell & Simon, 1972; Wickiegren, 1974). Certain of these seem more applicable to programming problems in general, children's programming in particular (Clements & Gullo, 1984; Lawler, 1985; Clement, Kurland, Mawby & Pea, 1986). The research reported in this paper was concerned with the teaching and learning of five such strategies – subgoals formation, forward chaining, systematic trial and error, alternative representation, and analogy. General definitions and explicit descriptions of each of these follow. Strategy descriptions are based on an extended version of Polya's (1973) description of general problem solving. They break the problem solving process into four parts – **problem definition**, specifying the goal and initial state of a problem, **plan development and implementation**, developing a plan and implementing a particular strategy, **evaluation**, checking to see that domain operators and general strategies have been correctly applied, and **recursion**, reapplying a strategy at a variety of levels.

Subgoals formation

Subgoals formation refers to breaking a single difficult problem into two or more simpler problems. Subgoals formation might thus be seen as the defining of a problem space. Even when no obviously solvable subgoals can be found, breaking a problem into its constituent parts makes its solution less formidable, more manageable, and less susceptible to errors. Subgoals formation can be described by the following four steps:

1. Problem definition. Specify the problem.

2. Subdivision. Examine the problem specification to see where it can be broken into smaller, self-contained problems. Specify these and their connections to the larger problem.

3. Evaluation. Test the subproblems generated for grain size and further decomposition. If the subproblems are manageable or cannot be further decomposed, solve them. Recombine these partial solutions into the total solution using the connections specified in step 2.

4. Recursion. Otherwise, repeat the second and third steps for each of the subproblems generated. Continue the process until no more smaller problems can be generated for any of the subproblems.

While subgoals formation might seem an obvious strategy to adults, it is not at all obvious to many children (Carver & Klahr, 1986). Moreover, of all the problem solving strategies, it can most clearly be implemented and concretized in Logo programming. In Logo, small subprocedures are easily written and placed in the workspace. Because these can be called from anywhere in a program, a program can simply be a list of such subprocedures, a very concrete representation of the subgoals that make up a programming solution.

Forward chaining

Forward chaining involves working from what is given in a problem towards the problem goal in step-by-step, transformational increments that bring one progressively closer to that goal. The forward chaining process can be decomposed into the following steps:

1. Problem definition. Specify the problem goal. Specify what is given. Specify the constraints, if any.

2. Transformation. Use domain operators to manipulate the givens to bring them closer to the goal state.

3. Evaluation. Compare the desired goal, the givens, and the transformation. Test to see whether the transformation is really closer to the goal than the givens. If it is not, redo step 2.

4. Recursion. Make the transformation a new given. Repeat steps 2 and 3 using the new given. Continue in this manner until the goal state is reached and the problem is solved.

A programming environment, especially an interpreted environment like Logo, is inherently supportive of the forward chaining process. Transformations can be implemented, their effects accessed, and successful changes instantiated as partial programs, with relative ease and little risk. A program can thus be developed in

incremental steps and such development provide a concrete model of the forward chaining process. An important part of forward chaining, however, involves the ability to choose appropriate transformations and evaluate whether or not these actually bring one nearer problem solution. Forward chaining thus requires some sort of mental model of the problem space, and is not, therefore typically a novice technique (Greeno & Simon, 1984).

Systematic trial and error

Systematic trial and error involves the recursive testing of possible solutions in a systematic, guided fashion, and the problem reduction and/or refinement resulting from such tests. The steps involved in the systematic trial and error process include:

1. Problem definition. Specify the problem goal.

2. Approximate solution. Create and implement a plan to solve the problem.

3. Evaluation. Compare the problem goal with the instantiated solution. If there are no discrepancies between them, the problem is solved. Otherwise, generate a description of the discrepancies between the desired goal and the instantiated solution.

4. Recursion. Use the description of goal/solution discrepancies to revise the plan, and reapply steps 2 and 3. Continue in this manner until the instantiated solution matches the desired goal.

Piaget (Ginsburg & Opper, 1980) believed that the application of systematic trial and error strategies was an important determinant of formal operational ability. Systematic trial and error, then, is an obvious candidate for testing Papert's (1980) notion that programming environments support the concretizing of the formal. Certain types of graphics programming, moreover, are paradigmatic of systematic trial and error strategies. Debugging also makes use of, and provides symbolic representations for, such techniques (Carver, 1987).

Alternative representation

Alternative representation involves conceptualizing a problem from differing perspectives. Polya (1973) writes that often the way a problem is stated is really all that makes it difficult, that simple restatement will make its solution obvious. Alternative representation is thus the antidote to functional fixedness (Dunker, 1945). It can be decomposed into the following four-step description:

1. Problem definition. Specify the problem.

2. Alternative representation. Generate an alternative problem specification

3. Evaluation. Test to see whether the new problem specification suggests problem solution. If it does, solve the problem.

4. Recursion. Otherwise, repeat steps 2 and 3 by generating and evaluating other problem specifications until a problem solution is found.

Programming is conducive to the development of alternative representations both because there are never single correct solutions to programming problems, and because differing representations can quite easily be instantiated and pragmatically tested in programming environments. Indeed, Clements and Gullo's (1984) study of the effects of Logo programming on young children's cognition found significant increases in their ability to produce alternative representations. Statz's (1973) finding of significant increases on permutation tasks may also support this view.

Analogy

Analogy involves the discovery of a particular similarity between two things otherwise more or less unlike, and "a mapping of knowledge from one domain (the base) onto another (the target) predicated on a system of relations that holds among the objects of both domains." (Gentner, 1987) An important factor in this process, especially in problem solving contexts, is goal-relatedness, how one domain is like another with respect to a specified goal (Holyoak & Koh, 1987). The use of analogy in problem solving can be decomposed into the following steps:

1. Problem definition. Specify the desired goal. Specify the base and the target systems.

2. Mapping. Perform a mapping between the base and target systems.

3. Evaluation. Test the soundness of the match in terms of both structural similarity and pragmatics (goal related conditions). If the analogy generated meets the goal conditions, and the structural similarity between the base and the target holds, the mapping is sound. Use the base domain solution to generate a solution in the target domain.

4. Recursion. Otherwise, return to step 1 and specify a new base domain. Apply steps 2 and 3 to it. Continue in this manner until an adequate representation is discovered.

Programming environments inherently support the development of analogy in that one is always mapping between computer code (a formal representation) and program output (a concrete representation). Indeed, Doug Clements (1987) found significantly better analogical reasoning among students with prior Logo experience.

These five problem solving strategies – subgoals formation, forward chaining, systematic trial and error, alternative representation, and analogy can be concretely represented, then, within a Logo programming context. We accordingly designed our instruction and our testing procedures around them. The instruction was split into units, one for each strategy. Each unit included first instruction focused on a particular strategy (declarative knowledge) followed by mediated practice solving problems designed to be particularly amenable to solutions employing that strategy (procedural knowledge). We likewise created six separate tests, each designed to measure students' facility in applying specific strategies to non-computing problems. Our goal was for students to transfer the strategies learned in the intervention to the paper and pencil tasks of the problem solving tests.

Methodology

Subjects

Subjects were one hundred students in the fourth through sixth grades of a private suburban elementary school. All subjects had at least one year (thirty hours) prior experience programming in Logo.

Procedure

All subjects were pre-tested on their ability to solve problems requiring the use of each of the five problem solving strategies under investigation, and randomly assigned by grade to one of three treatment conditions – a Logo graphics condition, a cut-paper manipulatives condition, or a discovery learning, Logo projects condition. Students in the first two conditions received the same basic problem solving instruction but differing practice environments. Students in the Logo graphics group received practice problems involving Logo graphics programming, while students in the cut-paper manipulatives group worked on similar problems involving the use of cut-paper manipulatives. Students in the third, Logo projects group received Logo programming problems to work on, but did not

receive direct problem solving instruction.

All subjects were post-tested upon completion of the intervention using different but analogous problem solving strategy tests. Differences between pre- and post-test scores were examined using analysis of variance with repeated measures. Independent variables were **test, strategy,** and treatment **group.** The dependent variables were the scores on the tests of each of the problem solving strategies. A more complete description of these tests and of each of the three treatment conditions follows.

Testing

Problem solving strategy tests consisted of sets of paper and pencil problems whose solutions required correct application of the particular strategies being investigated. Two different but analogous versions of each test were developed and randomly assigned by condition on the pre-test. Students were then assigned the alternative form of each test on the post-test. They were allowed as much time as they felt they needed to complete each test, but were required to work independently with no help from either the teachers or their peers.

Subgoals formation. (Figure 1) Our measure of students' subgoals formation ability consisted of mathematical word problems that required decomposition for correct solution. Students were asked not only to solve the problems but to show how they broke them into parts, and were given credit for correctly identified subgoals, as well as for the correct answer.

Forward chaining. (Figure 2) The test designed to measure subjects' forward chaining skills was a paper-and-pencil version of the computer program Rocky's Boots (The Learning Company, 1982). In Rocky's Boots, symbolic and, or, and not gates are combined to produce machines that respond to targeted attributes and sets of attributes (eg. blue diamonds, crosses or green circles, etc.). Combinations of gates must be built up in a forward chaining manner to achieve correct solutions. Our version had subjects draw the required connections.

Systematic trial and error. (Figure 3) Cryptography involves systematically trying and testing different symbol combinations to attain coherent decoding systems. We choose

two decoding exercises to test subjects' ability to systematically utilize trial and error strategies. The first of these was a shifted alphabet code. The second involved variations on a number code problem from Newell and Simon (1971).

Figure 1
Subgoals formation problems

Kathy went to the bookstore on Tuesday. She bought 2 notebooks that cost $1.50/each, 5 pencils that cost $.25/each, and a book that cost $12.50. How much did she spend altogether?

① 2(notebooks) × $1.50 = $3.00
② 5(pencils) × $.25 = $1.25
③ + $12.50 (book)
$16.75 (altogether)

John walks to the park each morning and rides the bus back home. It takes him 1 hour. The round trip bus ride takes 1/2 hour. How long would it take him to walk both ways?

① $2\sqrt{1/2} = 1/4$ bus one way
② $1 - 1/4 = 3/4$ walk one way
③ $2 \times 3/4 = 1\frac{1}{2}$ walk both ways

11

Figure 2
Forward chaining problems

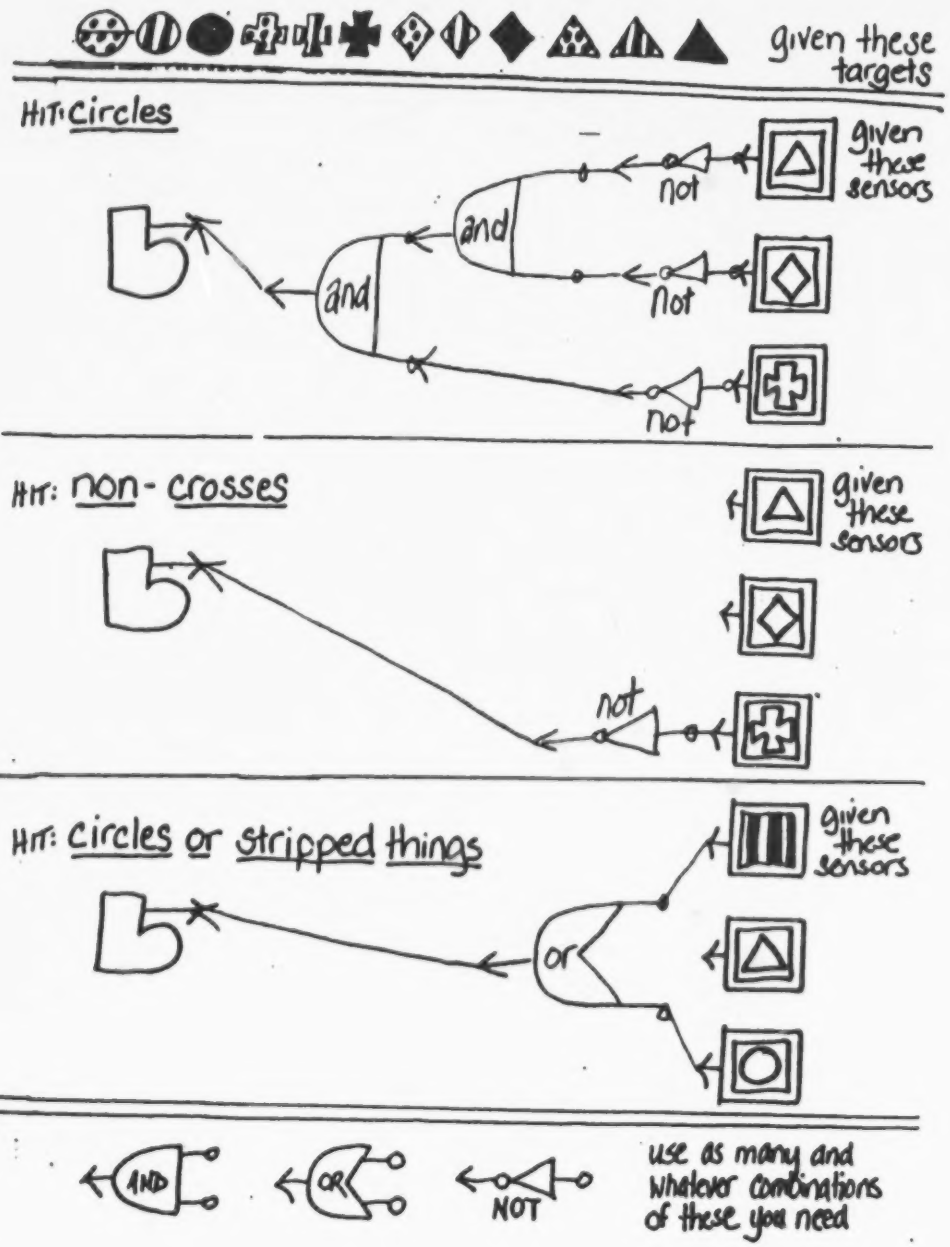


Figure 3
Systematic trial and error problems

"Al'k fgi gzwj mslsd lzw dzvq kafyk."

It's not over until the lady sings

This is a quotation you all know. It is written in a shifted alphabet, a code created by shifting the letters of the alphabet right or left (for example if h stood for a (BUT IT DOESN'T!!!), then i would stand for b, j for c, k for d, and so on.)

Figure out the code and decode the quotation. Show all your work, including the alphabet derivation and partial solutions.

a b c d e f g h i j k l m n o p
s t u v w x y z a b c d e f g h
q r s t u v w x y z
f j k l m n o p q r

1
5DONALD5
GERALD5
ROBERTO D=5

This is an addition problem written in code. Each letter stands for a number, but they are not in any spatial relationship to each other. The D stands for 5. Your job is to figure out the code and to give the problem in numbers. Show all your work including the derivations of the digits), 1, 2, 3, 4, 5, 6, 7, 8, 9 and partial solutions.

Alternative representation. (Figure 4) The measure of students' ability to create alternative representations we used was derived from the figures subtest of the Torrance Test of Creative Thinking (Torrance, 1972). Students were given sets of either parallel lines or circles and asked to use these as a basis for producing as many interesting and unusual drawings as they could. These were scored for quantity, diversity, originality, and elaboration.

Analogy. (Figure 5) Subjects' skill at analogical reasoning was measured with completion exercises comprised of items representing both verbal and visual analogy. They were given one analogy and asked to complete a second according to the relationship involved in the former.

Treatment

All subjects in all groups worked in pairs during their regular computer classes. A teacher and/or an intern were available for help on all problems. Both maintained a mediated learning approach toward student assistance, eliciting student and/or modeling their own cognitive processes as they guided students toward problem solution. Student pairs also helped each other solve problems. In general, classes met for two forty-five minute periods each week. The entire intervention took approximately two and one half months.

Direct instruction in each of the particular problem solving strategies was given to students in the Logo graphics and cut paper manipulatives conditions and were exactly the same for both groups. Wall charts based on the task analyses of the problem solving strategies but translated into children's language were made and used to introduce each strategy unit. Figure 6, for example, shows the chart for forward chaining. For each unit, the appropriate chart was produced and each step of the strategy explained and discussed. An example of how the strategy might be applied to help solve a problem was given and student examples elicited. The chart was then hung on the wall, problem sets distributed, and work on them begun.

Figure 4
Alternative representation problems

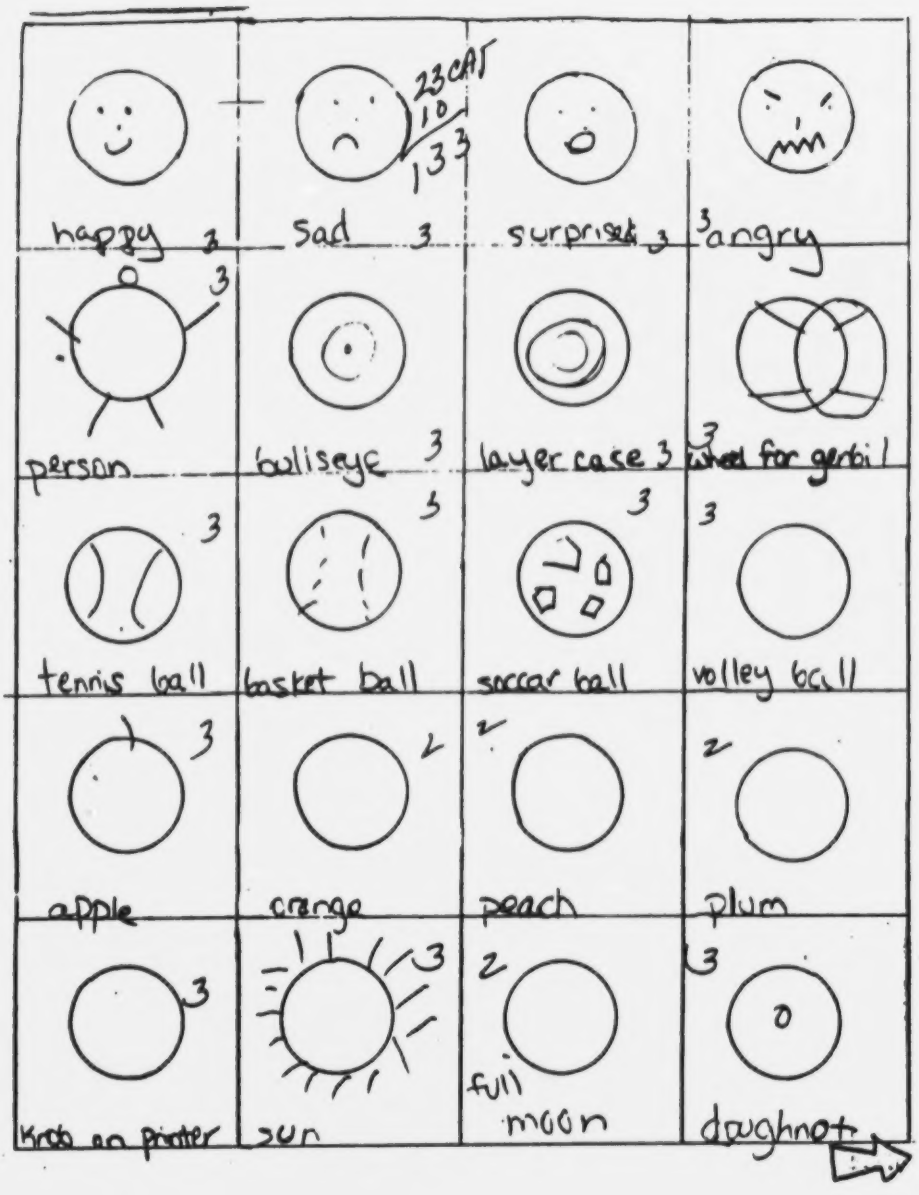


Figure 5
Analogy problems

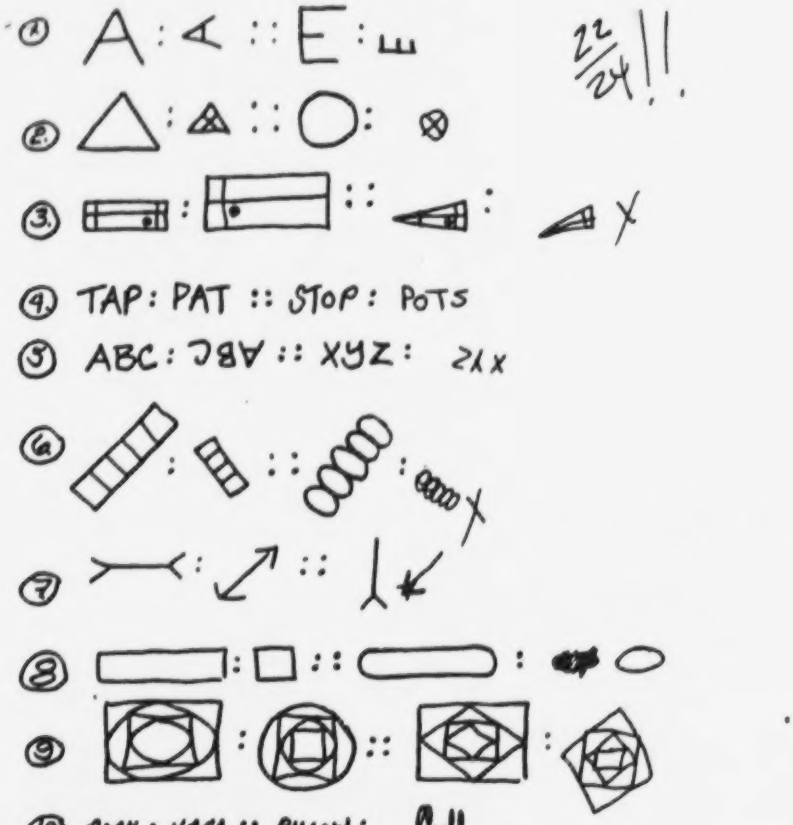

1. A : A :: E : E
22/24!!
2.
3.
X
4. TAP : PAT :: STOP : POTS
5. ABC : CBA :: XYZ : ZYX
6.
X
7.
8.
9.
10. ROCK : HARD :: PILLOW : fluffy
11. TALK : WALK :: MUTE : stop
12. CALF : COW :: PUPPY : dog
13. APPLE : ORANGE :: RED : orange
14. NOW : THEN :: PRESENT : past
15. LETTER : WORD :: SENTENCE : paragraph
16. FOOT : HAND :: TOE : fingers
17. PETAL : FLOWER :: PETAL : flower
18. CANDLE : CANDLE :: FLAME : FLAME

Figure 6
Wall chart for forward chaining strategies

FORWARD CHAINING

1. What is the problem?
   What is the problem goal?
   What is given in the problem?

2. How can you change what is given so it is more like the goal?
   Try it.

3. Is it really closer to the goal?
   If not, redo step 2.

4. If it is, make your change the new problem given. Go to steps 2 and 3 and redo for the new given. When the given matches the goal, you are done.

Students in the Logo graphics condition were given unit problem sets comprised of four graphics programming problems each and asked to solve them in Logo. Students in the cut-paper manipulatives condition were given unit sets comprised of four cut-paper manipulatives problems each, and provided with construction paper, rulers, scissors, and rubber cement with which to solve them. The purpose of the problems was to provide a practice environment in which students could evolve procedural representations of the problem solving strategies being taught. Thus, problems in each set were designed to be particularly amenable to solutions involving the use of the strategy under discussion, and difficult enough to be a genuine problem. Whenever possible, similar problems were used in both conditions. Figures 7, 8, and 9, for example, show subgoals formation problems for both conditions.

Figure 7

Subgoals formation problems: Logo graphics condition

SUBGOALS FORMATION -- GRAPHICS

1. House.

Put together a TRIANGLE and a SQUARE procedure to draw a HOUSE.

2. Neighborhood.

Put together many HOUSEs to draw a NEIGHBORHOOD.

3. Sailboat.

Use a TRIANGLE and a HALF.CIRCLE procedure to draw a SAILBOAT.

4. Face.

Put together various shapes of your own choosing to draw a FACE.

Figure 8

Subgoals formation problem #1: Cut-paper manipulatives condition

SUBGOALS FORMATION - 1

Cut a triangle and a square out of construction paper. Put them together to make a house. Paste your house here.

Figure 9

Subgoals formation problem #2: Cut-paper manipulatives condition

SUBGOALS FORMATION - 2

Make many houses from triangles and squares. Paste them together here to make a neighborhood.

In some cases, however, the use of problems with similar surface features would result in highly disparate degrees of cognitive difficulty. The Logo graphics problems in the unit on systematic trial and error, for example, had students perform a variety of screen formatting activities – drawing a double border around the screen, a target, an aerial view of city blocks, and a house plan – which required the testing and refining of Logo procedures. Such activities could be done much easier with cut paper. Pentamino puzzles which likewise required the testing and refining of possible solutions, but were of similar cognitive difficulty, were therefore substituted.

For each problem they solved, students in both groups were required to fill out a problem solving worksheet that showed the givens, the goal, and the solutions steps for that problem. Figures 10 and 11 show examples of completed worksheets. Students in the Logo graphics condition were also required to turn in a listing and a run of their programs. Students in the cut-paper manipulatives condition were required to turn in their completed designs.

Students in the Logo projects condition were not given direct problem solving strategy instruction and were not required to fill in problem solving worksheets. They worked on Logo programming projects they selected from lists covering four areas of programming concepts – procedures, variables, conditionals, and recursion. Projects involved both graphics and list manipulation problems and were chosen to represent the range of projects typically assigned in Logo classes. Examples of these are given in Figures 12 and 13. Students in this condition were required to work on one or more project from each list, progressing through the lists at their own speed and as time allowed. Their projects could be as simple or as complex, and utilize whatever programming and/or problem solving strategies they desired. Just like students in the Logo graphics condition, students in the Logo projects condition were required to turn in a listing and a run of their programs.

Figure 10

Problem solving worksheet for subgoals formation problem #1:

Logo graphics condition

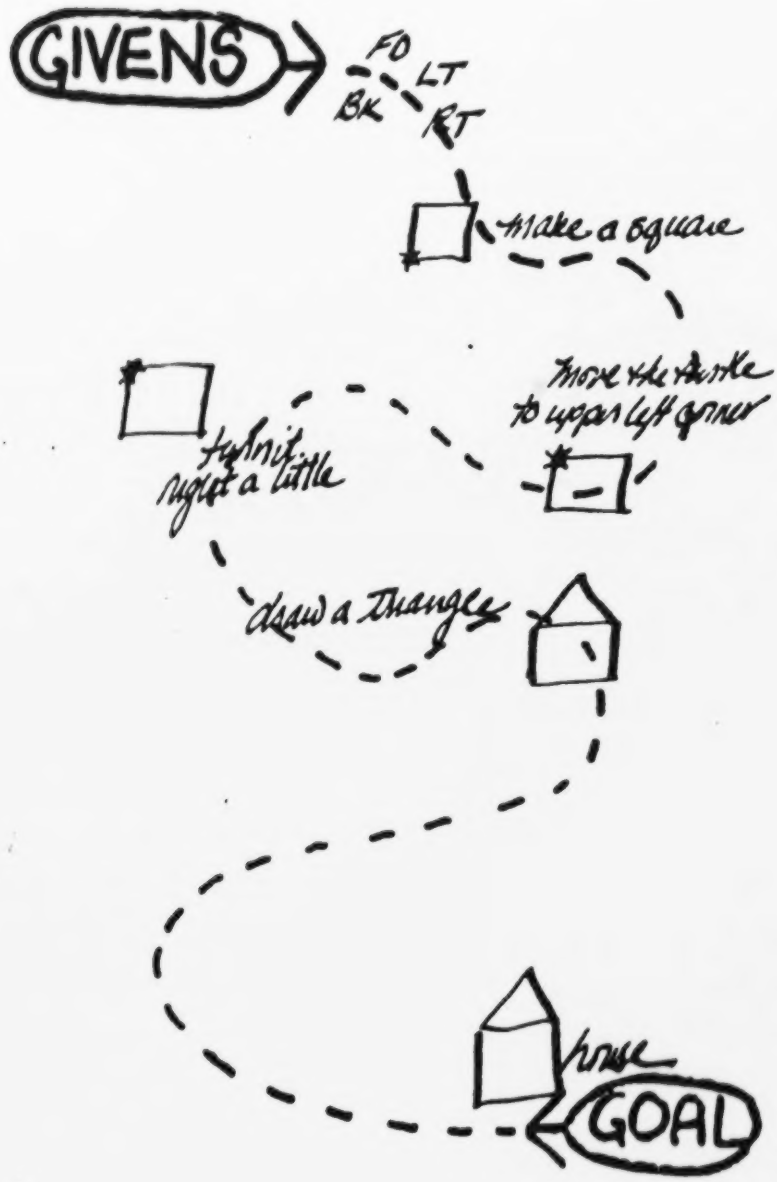


Figure 11

Problem solving worksheet for subgoals formation problem #1:

Cut-paper manipulatives condition

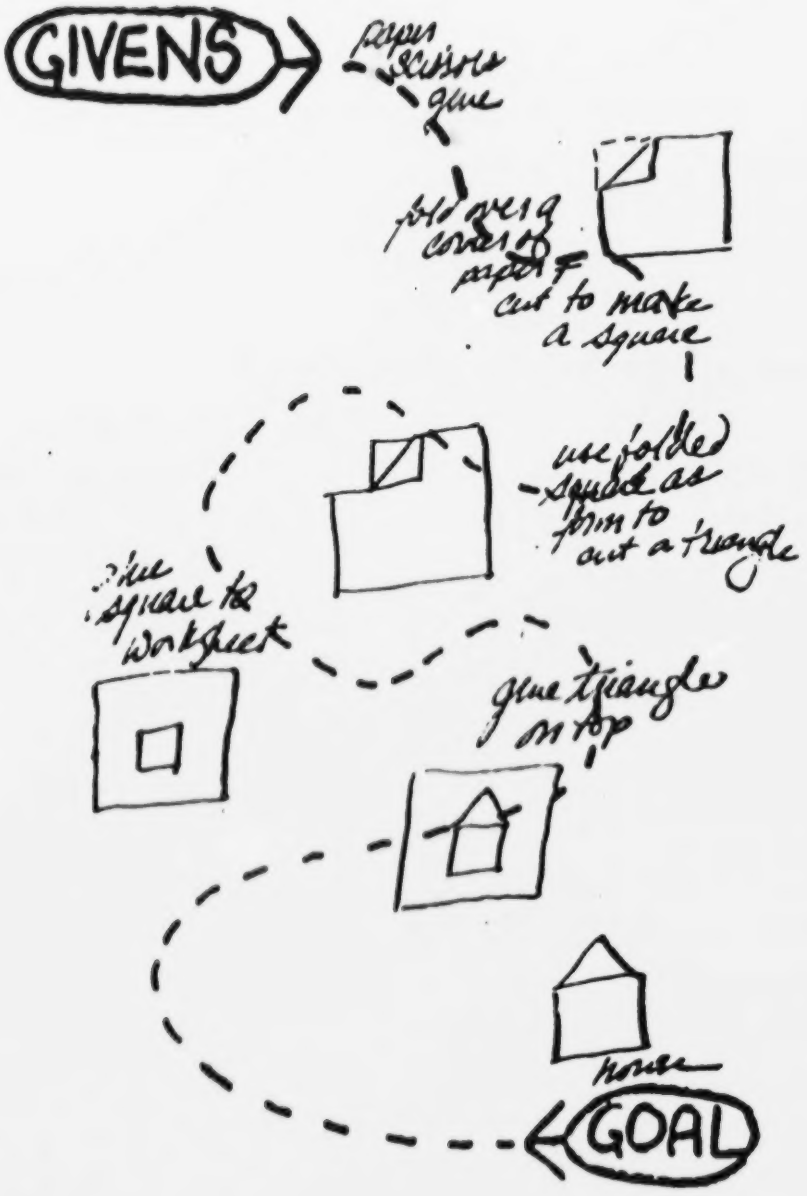


Figure 12

Projects dealing with procedures:

Logo projects condition

LOGO PROJECTS

1. Write procedures that draw regular shapes. Put the shapes together in a superprocedure that draws a picture or a design.

2. Write procedures that draw the parts of a face (i.e. EYE, LIPS, NOSE, EAR). Put them together in a superprocedure that draws a FACE.

3. Write a procedure that PRINTs a poem. Include it in a superprocedure that illustrates your poem.

4. Write procedures that draw all the letters in your name. Put them together in a superprocedure that writes your name on the screen.

5. Write a superprocedure that humorously illustrates an idiom (i.e. "His head was in the clouds", "You drive me up a wall"). Include animation in your idiom illustration.

6. Write a superprocedure that draws a sign or a poster. Use letters and drawing in it.

Figure 13

Projects involving variables:

Logo projects condition

PROJECTS WITH VARIABLES

1. Write procedures to draw variable sized shapes.

2. Write procedures to draw variable sized shapes filled with whatever color the person wants.

3. Write a procedure that writes a name the person types in all over the screen.

4. Write a MADLIBS procedure that asks the person to type in various words (i.e. an adjective, a name, a place, a verb, etc) and then prints out a funny story using those words.

5. Write a procedure that draws a variable-sized house.

6. Write procedures to create a variable sized alphabet.

Students were tested before and after treatment on measures of each of the five problem solving skills. Two different but analogous versions of each test were developed. Students were randomly assigned one or the other version of each test by condition on the pre-test, and then were given the alternative version of each test on the post-test to assure pre- to post-test reliability. Mean pre-test scores were compared between groups using one-way analysis of variance and found to be statistically equivalent ($F_{2,97} = 0.33$, $p > .10$), hence the groups were assumed to be generally equal in problem solving ability before treatment.

Raw scores on all tests except those for alternative representation were converted to percent correct scores and compared using three-way analysis of variance with repeated measures. Because they had no maximum possible scores and so no percentage correct could be calculated for them, alternative representation measures were evaluated seperately using two-way analysis of variance with repeated measures.

Results

The results of the various analyses argue that explicit instruction and mediated Logo programming practice is superior to both similar instruction with cut-paper manipulatives practice, and discovery learning within Logo programming domains for supporting the aquisition and transfer of subgoals formation, forward chaining, systematic trial and error, and analogy strategies. Within this context, it appears that such combination is most supportive of the teaching and learning of subgoals formation strategies among students in the age groups studied. Results involving the teaching and learning of alternative representation strategies were more problematical and require further investigation.

Subgoals formation, forward chaining, systematic trial and error, and analogy

Means and standard deviations for the three-way analysis of variance comparing scores on measures of subgoals formation, forward chaining, systematic trial and error, and analogy are given in Tables 1 and 2. The resulting ANOVA table is shown in Table 3.

Table 1

Means Table for subgoals formation (SG), forward chaining (FC), systematic trial and error (STE), and analogy (ANAL)

		GRAPHICS	CUT-PAPER	PROJECTS	(T X S)
SG	PRE	44.0	35.6	29.0	36.0
	POST	63.1	41.6	28.4	43.9
FC	PRE	58.9	51.5	55.6	55.3
	POST	65.9	46.1	50.7	54.0
STE	PRE	36.8	20.6	26.5	27.8
	POST	48.1	17.5	26.0	30.3
ANAL	PRE	78.9	75.8	81.5	78.8
	POST	86.1	75.5	79.5	80.3
(GROUP)		60.2	45.5	47.1	50.8

Table 2

Standard Deviations for subgoals formation (SG), forward chaining (FC), systematic trial and error (STE), and analogy (ANAL)

		GRAPHICS	CUT-PAPER	PROJECTS
SG	PRE	22.5	22.1	24.5
	POST	22.5	29.7	25.8
FC	PRE	18.1	15.2	18.8
	POST	17.0	15.8	20.6
STE	PRE	25.0	27.7	27.9
	POST	21.8	27.4	27.2
ANAL	PRE	11.4	14.1	12.4
	POST	8.6	14.5	16.0

Table 3

ANOVA Table for subgoals formation (SG), forward chaining (FC), systematic trial and error (STE), and analogy (ANAL)

	SS	DF	MS	F	P
MEAN	2074999.0	1	2074999.0	1568.08	0.0000
GROUP	33907.5	2	16953.7	12.81	0.000
ERROR	128357.9	97	1323.3		
TEST	1576.6	1	1576.6	5.94	0.0166
TG	6871.7	2	3437.3	12.96	0.0000
ERROR	25725.0	97	265.2		
STRATEGY	283190.9	3	94397.0	207.11	0.0000
SG	13478.5	6	2246.4	4.93	0.0001
ERROR	132630.9	97	455.8		
TS	2274.2	3	758.1	3.64	0.0132
TSG	743.4	6	123.9	0.6	0.7338
ERROR	60540.8	97	208.0		

The independent variables in the research design were test, strategy, and treatment group. The dependent variables were the scores on the tests of each of the four problem solving strategies submitted for analysis. There was one between-subjects factor, treatment group, and two within-subjects factors, test and strategy. Significant main effects were found for all these factors (group, $F_{2,97} = 12.81$, $p < .01$; test, $F_{1,97} = 5.94$, $p < .05$; strategy, $F_{3,97} = 207.11$, $p < .01$), indicating significant differences along all these dimensions. Of these, only the group effect is particularly meaningful. Because the groups were statistically equivalent before, but not after receiving the interventions, the group effect indicates differences in scores resulting from treatment. This result favors the Logo graphics condition which had an overall mean score of 60.2 percent correct, compared with the cut-paper manipulatives group whose mean score was 45.5 percent correct, and the Logo projects group whose mean score was 47.1 percent correct.

Four different interaction effects were also examined by this design.

Significant **test by group**, ($F_{2,97} = 12.96$, $p < .01$), **test by strategy** ($F_{3,97} = 3.64$, $p < .05$), and **strategy by group** ($F_{3,97} = 207.11$, $p < .01$) interactions were found. No **test by strategy by group** interaction was discovered ($p > .10$).

The interaction of greatest interest is test by group. It indicates differences in pre- to post-test changes in scores resulting from the differing treatments. The tests by group interaction was examined in greater detail by assessing the **simple test effects** at each level of **group**. A strong test effect was found for the Logo graphics group ($F_{1,97} = 29.95$, $P < .01$), indicating significant pre- to post-test changes among students receiving that treatment, but not for the other two groups ($p > .10$). Table 4 shows the mean differences between pre- and post-test scores by group and strategy. Marginal group means reveal that students in the Logo graphics group improved an average of 11.1 percentage points on the four measures, while the scores of students in the cut-paper manipulatives group remained essentially the same, and the scores of students in the Logo projects group actually declined slightly (although not significantly). These results argue strongly that the Logo graphics intervention, and the Logo graphics intervention alone, resulted in improvements in students' problem solving abilities. It is interesting to note that mean pre- to post-test increases among students receiving the Logo graphics treatment were nearly identical to those observed in our pilot research indicating a consistant treatment effect, and adding support to conclusions arguing for the efficacy of the intervention.

Table 4

Mean Pre- to Post-Test Changes for subgoals formation (SG), forward chaining (FC), systematic trial and error (STE), and analogy (ANAL)

	GRAPHICS	CUT-PAPER	PROJECTS	(STRATEGY)
SG	19.1	6.0	-0.6	7.9
FC	7.0	-5.4	-4.9	-1.3
STE	11.3	-5.4	-0.5	2.5
ANAL	7.2	-3.1	-2.0	1.5
(GROUP)	11.1	-0.7	-2.0	2.7

Because the various problem solving strategy measures were not designed to be equivalent, the two interaction effects involving the strategy dimension are not necessarily meaningful. The test by strategy interaction is the more interesting of the two. It indicates that students had greater pre- to post-test changes on certain strategy measures than on

others, but, because the measures were not equivalent, any comparison of mean differences across strategies is problematical. Looking at the mean differences table with this in mind, notice that students showed by far the greatest increases on subgoals formation tests, and that part of the reason for this is that students in the Logo graphics group improved on these measures, whereas neither they, nor students in the Logo projects group improved on any of the others. An examination of the **simple test effect** at each level of **strategy** reveals that indeed subgoals formation measures were the only ones on which the majority of students exhibited significant pre- to post-test differences ($F_{s,97} = 11.59$, $p < .01$) Such findings at least suggest that subgoals formation strategies were more easily acquired by students in this age group, a finding which concurs with the results of our pilot research.

The strategy by group interaction indicating differing scores relative to students in other groups on differing strategy measures. Were thos measures equivalent, or were patterns of differences found on the pre-test radically different from overall patterns, then the differing efficacies of particular treatments for supporting the acquisition and transfer of specific strategies could be argued. Neither, however, was the case. The result, then, is significant, but not meaningful.

Alternative representation

The tests of students' ability to create alternative representations had no maximum possible correct. Total scores on these tests ranged from a low of 21 to highs of over 250 points, thus the variance on this measure was very large. The problem was compounded by the facts that students in all groups showed improvements on this measure, and that a comparison of pre-test group means reveals they were not statistically equivalent ($F_{2,97} = 4.99$, $p < .01$), hence that the groups were equal in alternative representational ability before treatment. Means and standard deviations of raw scores on these tests are given in Tables 5 and 6. The analysis of variance for alternative representation is given in Table 7.

Table 7

Means Table for alternative representation

	GRAPHICS	CUT-PAPER	PROJECTS	(TEST)
PRE	70.7	106.2	83.3	86.8
POST	109.1	123.1	94.9	108.7
(GROUP)	89.9	114.7	89.9	97.8

Table 6

Standard Deviations for alternative representation

	GRAPHICS	CUT-PAPER	PROJECTS
PRE	30.2	58.3	45.0
POST	43.4	53.9	53.5

Table 7

ANOVA Table for alternative representation

(N = 100)

	SS	DF	MS	F	P
MEAN	1913259.6	1	1913259.6	563.35	0.0000
GROUP	28058.5	2	14029.3	4.13	0.0000
ERROR	329431.8	97	3396.2		
TEST	24819.1	1	24819.1	10.55	0.0016
TG	6631.9	2	3315.9	2.57	0.0821
ERROR	125363.8	97	1292.4		

The analysis of variance for tests of alternative representation reveals significant main effects for group ($F_{2,97} = 4.13$; $p < .05$) and test ($F_{1,97} = 19.20$; $p < .01$). Neither of these are particularly meaningful in themselves. The group effect is not meaningful because the groups were not equivalent to begin with. The test effect does indicate significant overall pre- to post-test differences, but the means table reveals that students in all groups improved on this measure. Because all groups showed improvments, what is not and cannot be known is whether such increases represent genuine learning or are rather the result of practice and/or maturation.

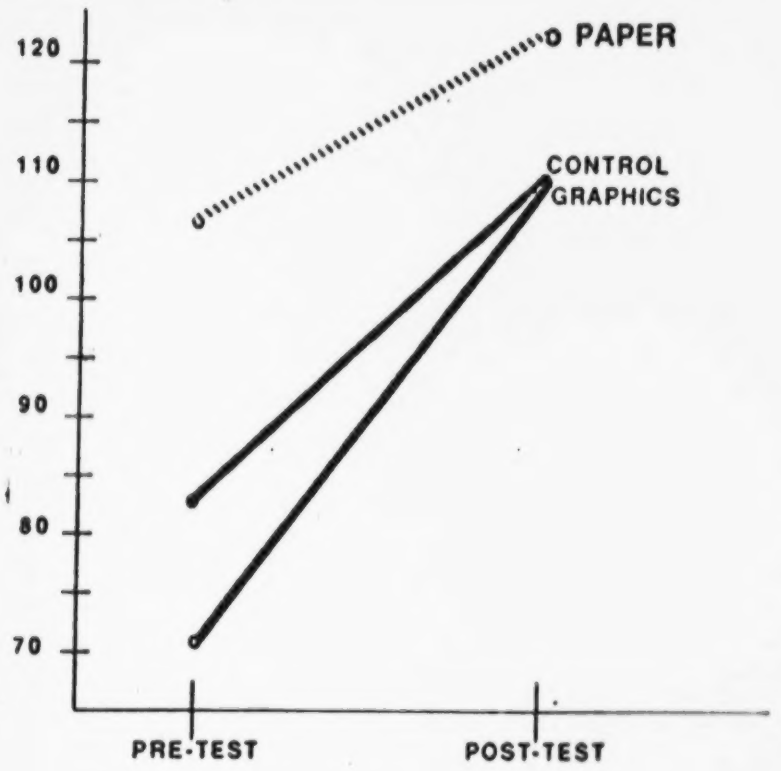
What would be meaningful would be a solid tests by group interaction effect. Unfortunately the analysis of variance reveals only weak interaction ($F_{2,97} = 2.57$; $.05 > p < .10$). To examine the test by group interaction in greater detail, the simple test effect was assessed at each level of group. A strong test effect was found for the Logo graphics group ($F_{1,97} = 18.91$; $p < .01$), whereas only a weak test effect was found for the cut-paper manipulatives group ($F_{1,97} = 3.61$, $.05 > p < .10$), and no test effect at all was found for the

Logo projects group ($F_{1,97} = 1.81$; $p > .10$). It is interesting to note that these results mirror the findings for subgoals formation and perhaps indicate that direct instruction was a more significant factor in the success of the intervention than Logo programming, although both appear necessary for transfer to take place.

These differences are illustrated by Figure 14 which shows the mean pre- to post-test differences for each group on alternative representation measures. It can be seen that the Logo graphics group showed improvements nearly twice as great as those of the group with the next highest improvements, the cut-paper manipulatives group. However, because students in the Logo graphics group had much lower pre-test scores than students in the other two groups, the greater gains they made might be accounted for by differential ability levels as well as by treatment effects. The most we can conclude, then, is that it is possible that students in the Logo graphics group showed an increased facility for alternative representation, and that such possibility argues for further investigation with more evenly matched groups.

Figure 14

Mean scores for alternative representation by treatment groups



Discussion

In terms of our research questions, we can conclude that, direct instruction was more effective than discovery learning in supporting the acquisition and transfer of problem solving skills from Logo programming to non-computing domains, and that the Logo programming envrionment was an improtant factor in such acquisition and transfer. The results argue quite strongly, then, for the superiority of direct instruction and mediated Logo programming practice over both similar instruction with cut-paper manipulatives practice, and discovery learning in similar paractice environments, for the acquisition and transfer of four problem solving strategies – subgoals formation, forward chaining, systematic trial and error, and analogy – among middle school students. Indications are that such instruction and practice may likewise be most effective for the teaching and learning of alternative representation measures, although further research with more evenly matched groups is needed to determine its effectiveness in this area.

Such findings argue that the intervention we designed does, in fact, support the teaching and learning of problem solving, that increased scores on problem solving strategy measures resulted from it and not from the effects of practice and/or maturation. They also lend support to our analysis of the Logo/problem solving literature, in that direct instruction and mediated programming practice resulted in students' acquisition and transfer of problem solving skills whereas Logo programming practice alone did not. Indeed, indications are that direct instruction may be a more important factor than Logo programming in the success of the intervention we designed. Both factors, however, appear necessary, as students in the group receiving direct instruction with cut-paper manipulatives practice fared little better than students in the Logo discovery group.

Two issues raised by the research results deserve further comment. The involve the efficacy of knowledge-based instructional design for the teaching and learning of problem solving, and the mediational nature of computer programming environments, the Logo programming environment in particular.

Knowledge-based instructional design refers to premising the design of instruction on desired knowledge outcomes rather than on desired behavioral outcomes. The distinction is a real one. The desired outcome of problem solving instruction, for example, is increased problem solving abilities. When such abilities are conceived in terms of desired behaviors, they are understood as being able to solve particular kinds of problems and are not broken down any further because they are not conceptualized beyond this behavioral level. Problem solving ability is seen as its behavioral manifestation, hence, the prescribed

instruction has correspondingly involved practice solving such problems. Little emphasis is placed on the general knowledge structures which underlie their solution, and the particulars of specific strategies are not addressed. This sort of instruction, especially in the context of computer programming, has not been successful in increasing students' problem solving abilities (Abbott, Salter & Soloway, 1986; Shaw, 1986; Patterson & Smith, 1986; Mandinach & Linn, 1987). They are not successful because complex cognitive behaviors like problem solving involve more than their manifest behaviors and must be addressed at a deeper level, at the level of the knowledge structures which support such behaviors.

Indeed when one conceives of problem solving instruction in terms of knowledge outcomes, the desired outcome is understood as the knowledge necessary to solve particular kinds of problems. The focus is not on the behavior but on the knowledge supporting the behavior. The knowledge supporting problem solving behviors is the procedural knowledge of the specific steps involved in particular problem solving strategies. That such knowledge underlies problem solving has been demonstrated by problem solving computer programs (Newell & Simon, 1972; Anderson, 1983). Moreover, such knowledge has a declarative as well as a procedural component (Anderson, 1985; Flavell, 1985). At least in the case of the instruction we designed, a direct focus on a declarative knowledge of the steps involved in the particular problem solving strategies was a necessary factor in the success of the intervention. Because it focuses on behaviors and not knowledge, behvaiorally-based instructional design ignores this important, perhaps critical, declarative knowledge component.

Knowledge-based instructional design played a critical role in the success of the instructional model we developed. It may well be a more useful approach to the design of problem solving instruction in general, perhaps to the design of any instruction concerned with complex cognitive behaviors. It clearly deserves further careful study.

The mediational nature of computing environments refers to the way in which computers can be used to support what Papert (1980) refers to as "transitional objects to think with." Papert maintains that computing environments can support quasi-concrete, dynamic representations of abstract ideas, representation that can be manipulated and tested and which provide immediate concrete feedback concerning the soundness of their formulation. Such representations are transitional in that they can help bridge the gap between condrete and formal thought. They are mediational in that they support abstract thinking which might otherwise overwhelm working memory.

Such a view is supported by the finding that students given direct problem solving instruction and mediated practice in a non-computing envrionment did not learn the problem solving strategies as well as did students given similar direct instruction and mediated Logo

practice. Lehrer and Randle's work (1987) also suggests such a view. If computing environments can be designed to support such transitional objects for thinking, they might play an important role in education. The notion certainly deserves further investigation.

The development of problem solving and critical thinking skills is a crucial problem for education today. The research presented in this paper clearly demonstrates a successful model for developing particular problem solving abilities among upper elementary student populations, a model which, in itself, deserves further study. More importantly, it suggest methods for designing instruction that might develop such skills in a broad range of subject area contexts, in particular, knowledge-based instructional design and the mediational use of computing environments. In today's educational climate, such methods deserve immediate serious attention.

References

Anderson, J. R. (1983) *The Architecture of Cognition.* Cambridge, MA: Harvard University Press.

Anderson, J. R. (1985) *Cognitive Psychology and its Implications.* NY: W. H. Freeman.

Carver, S. M. (1987) Transfer of Logo debugging skill: analysis, instruction, and assessment. *Computer Systems Group Bulletin, 14* (1) , 4-8.

Carver, S. M. and Klahr, D. (1986) Assessing children's Logo debugging skills with a formal model. *Journal of Educational Computing Research, 2* (4), 487-525.

Clement, C. A., Kurland, D. M., Mawby, R. and Pea, R. D. (1986) Analogical reasoning and computer programming. *Journal of Educational Computing Research, 2* (4), 73-94.

Clements, D. H. (1987) Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. *Journal of Educational Computing Research, 3* (1), 94-137.

Clements, D. H. and Gullo, D. F. (1984) Effects of computer programming on young children's cognitive abilities and achievement. *Journal of Educational Psychology, 76* , 1051-1058.

Flavell, J. H. (1985) *Cognitive Development.* Englewood Cliffs, NJ: Prentice-Hall.

Gentner, D. (1987) *Mechanisms of Analogical Learning.* Urbana, IL: University of Illinois, Department of Computer Science.

Ginsburg, H. and Opper, S. (1980) *Piaget's Theory of Intellectual Development.* Englewood Cliffs, NJ: Prentice-Hall.

Gorman, H., Jr. and Bourne, L. E. (1983) Learning to think by learning Logo: rule learning in third-grade computer programmers. *Bulletin of the Psychonomic Society, 21*, 165-167.

Greeno, J. G. and Simon, H. A. (1984) *Problem Solving and Reasoning.* (Technical Report No UPITT/LRDC/ONR/APS-14). Washington, DC: Learning Research and Development Center, Office of Naval Research.

Holyoak, K. J. and Koh, K. (1987) Surface and structural similarity in analogical transfer. *Memory and Cognition, 15*, 332-340.

Johanson, Roger P. (1988) Computers, cognition and curriculum: retrospect and prospect. *Journal of Educational Computing Research, 4* (1), 1-30.

Lawler, R. W. (1985) *Computer Experience and Cognitive Development: A Child's Learning in a Computer Culture.* New York: Halsted.

Lehrer, R. and Randle, L. (1987) Problem solving, metacognition and composition: the effects of interactive software for first-grade children. *Journal of Educational Computing Research, 3* (4), 408-428.

Lehrer, R., Sancilio, L. and Randle, L. (1988) Learning pre-proof geometry with Logo. Paper presented at the annual meeting of the American Educational Research Association, New Orleans.

Leron, U. (1985) Logo today: vision and reality. *The Computing Teacher, 12* (6), 26-32.

Mandinach, E. B. and Linn, M. C. (1987) Cognitive consequences of programming: achievements of experienced and talented programmers. *Journal of Educational Computing Research, 3* (1), 53-72.

Newell, A. and Simon, H. A. (1972) *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Papert, s. (1980) *Mindstorms*. New York: Basic Books.

Papert, S., Watt, D., diSessa, A., and Weir, S. (1979) *Final Report of the Brookline Logo Project*, (Logo Memo 53). Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Patterson, J. H. and Smith, M. S. (1986) The role of computers in higher order thinking. In J. A. Culbertson and L. L. Cunningham (Eds.) *Microcomputers and Education*. Chicago: University of Chicago Press.

Pea, R. D. and Kurland, D. M. (1984) On the cognitive effects of learning computer programming *New Ideas in Psychology, 2* (2), 137-167.

Pea, R. D. and Kurland, D. M. (1987) Logo programming and the development of planning skills. In K. Sheingold and R. D. Pea (Eds.) *Mirrors of Minds*. Norwood, NJ: Ablex.

Polya, G. (1973) *How To Solve It*. Princeton, NJ: Princeton University Press.

Salomon, G. and Perkins, D. N. (1987) Transfer of cognitive skills from programming: when and how? *Journal of Educational Computing Research, 3* (2), 149-170.

Shaw, D. G. (1986) Effects of learning to program a computer in BASIC or Logo on problem solving abilities. *AEDS Journal, 19* (2/3), 176-189.

Statz, J. (1973) *Problem Solving and Logo: Final Report of the Syracuse Logo Project*. Syracuse, NY: Syracuse University.

Swan, K. and Black, J. B. (1988) The cross-contextual transfer of problem solving skills from computing to non-computing domains. Paper presented at the annual meeting of the American Educational Research Association, New Orleans.

Thompson, A. D. and Wang, H. M. C. (1988) Effects of a Logo microworld on student ability to transfer a concept. *Journal of Educational Computing Research, 4* (3), 335-347.

Torrance, E. P. (1972) *Torrance Test of Creative Thinking*. Lexington, MA: Personal Press.

Wickelgren, W. A. (1974) *How To Solve Problems*. San Francisco: W. H. Freeman.